REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
2002	Journal Article	2000 to 2002

4. TITLE AND SUBTITLE
Development of a Testbed for Distributed Satellite Command And Control

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62601F

6. AUTHOR(S)
Paul Zetocha
Margarita Brito*

5d. PROJECT NUMBER
8809

5e. TASK NUMBER
LJ

5f. WORK UNIT NUMBER
01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/VS
3550 Aberdeen Ave. SE
Kirtland ARB, NM 87117-5776

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; Distribution unlimited.

13. SUPPLEMENTARY NOTES
*Princeton Satellite Systems

14. ABSTRACT

At the Air Force Research Laboratory's Space Vehicles Directorate we are investigating and developing architectures for commanding and controlling a cluster of cooperating satellites through prototype development for the TechSat-21 program. The objective of this paper is to describe a distributed satellite testbed that is currently under development and to summarize near term prototypes being implemented for cluster command and control. To design, develop, and test our architecture we are using eight PowerPC750 VME-based single board computers, representing eight satellites. Each of these computers is hosting the OSE™ real-time operating system from Enea Systems. At the core of our on-board cluster manager is ObjectAgent. ObjectAgent is an agent-based object-oriented framework for flight systems, which is particularly suitable for distributed applications. In order to handle communication with the ground as well as to assist with the cluster management we are using the Spacecraft Command Language (SCL). SCL is also at the centerpiece of our ground control station and handles cluster commanding, telemetry decommutation, state-of-health monitoring, and Fault Detection, Isolation, and Resolution (FDIR). For planning and scheduling activities we are currently using ASPEN from NASA/JPL. This paper will describe each of the above components in detail and then present the prototypes being implemented.

15. SUBJECT TERMS
prototype, TechSat-21, satellite testbed, ObjectAgent, spacecraft command language

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Paul Zetocha
Unclassified	Unclassified	Unclassified	Unlimited	7	19b. TELEPHONE NUMBER *(include area code)* (505) 853-4114

082

Development of a Testbed for Distributed Satellite Command and Control

Paul Zetocha
Space Vehicles Directorate
Air Force Research Laboratory
Paul.Zetocha@Kirtland.af.mil

Margarita Brito
Princeton Satellite Systems
megui@psatellite.com

Abstract — At the Air Force Research Laboratory's Space Vehicles Directorate we are investigating and developing architectures for commanding and controlling a cluster of cooperating satellites through prototype development for the TechSat-21 program. The objective of this paper is to describe a distributed satellite testbed that is currently under development and to summarize near term prototypes being implemented for cluster command and control. To design, develop, and test our architecture we are using eight PowerPC750 VME-based single board computers, representing eight satellites. Each of these computers is hosting the OSE™ real-time operating system from Enea Systems. At the core of our on-board cluster manager is ObjectAgent. ObjectAgent is an agent-based object-oriented framework for flight systems which is particularly suitable for distributed applications. In order to handle communication with the ground as well as to assist with cluster management we are using the Spacecraft Command Language (SCL). SCL is also at the centerpiece of our ground control station and handles cluster commanding, telemetry decommutation, state-of-health monitoring, and Fault Detection, Isolation, and Resolution (FDIR). For planning and scheduling activities we are currently using ASPEN from NASA/JPL. This paper will describe each of the above components in detail and then present the prototypes being implemented.

TABLE OF CONTENTS

1. INTRODUCTION

For many satellite missions large monolithic satellites are cases this results in costly satellites which are more complex, more susceptible to failure, and which have performance characteristics that are less than optimal due to realistic physical size limitations. Recently various organizations have begun to explore how distributed clusters of cooperating satellites can replace their larger monolithic counterparts resulting in an overall cost reduction, enhanced mission performance, and increased system fault tolerance. Large clusters of satellites flying in formation are required to have some level of on-board autonomy in order to: fly within specified tolerance levels; perform collision avoidance; perform FDIR; and plan and schedule activities. In addition, from an operations standpoint commanding and controlling a large cluster of satellites can be very burdensome for ground operators. We are addressing these issues by incorporating an on-board cluster manager which will in essence provide the capability to treat the cluster of satellites as a single virtual satellite. From a ground perspective the ground control station must also be able to treat the cluster as a virtual satellite.

Several prototypes are planned in order to develop, test, and demonstrate cluster management functionality and the virtual satellite concept. The initial prototype being developed involves three satellites flying in formation with the ability to perform autonomous reconfiguration based on equipment failure. Among other tasks, on a periodic basis the on-board cluster manager accumulates telemetry from all satellites and forwards that, along with cluster level status, to the ground for display, monitoring, and archiving. The ground station has the ability to send either commands to individual satellites or to the cluster manager. In the latter case the cluster manager uses on-board knowledge to determine the appropriate response. The above functionality as well as others will be elaborated on in the sections that follow.

2. TESTBED ARCHITECTURE

In order to develop and test the various satellite cluster command and control techniques, as well as other TechSat-21 technologies, a distributed satellite testbed is being developed. A simplified version of the TechSat-21 testbed is depicted in figure 1.

required to satisfy objectives to a satisfactory level. In many

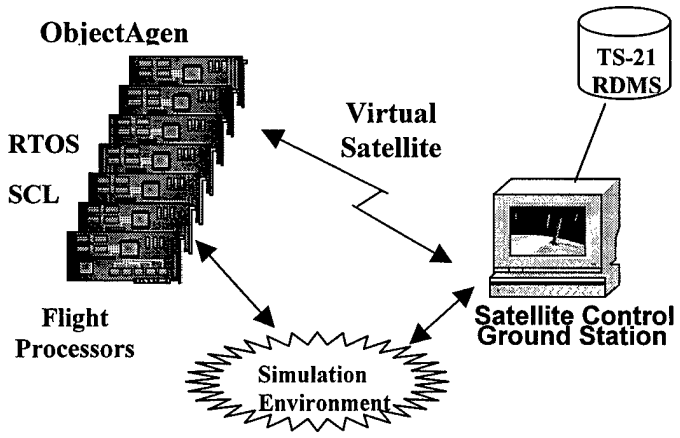


Figure 1: TechSat-21 testbed

The figure shows the three major components of the testbed. The ground system is shown on the right side, the flight system is shown on the left side and the simulation environment is in the middle. The Realt-Time Operating System (RTOS) used is OSE and the Remote Data Base Management System (RDMS) is NT SQL Server. The first two components will be discussed in greater detail in sections 2.1 and 2.2 respectively. The simulation environment will not be discussed because different simulations can be used depending on the required level of fidelity.

2.1 GROUND SYSTEM

Commanding and controlling a cluster of satellites from the ground pose many challenges. In order to optimize ground operation cost and manpower, different methods are needed to command and control the cluster, monitor the cluster, and to perform telemetry decommutation. These first two tasks are highlighted here. The details of cluster telemetry decommutation are left for discussion elsewhere.

For a large cluster it is not efficient or cost effective to command satellites on an individual basis. A more efficient method is to send commands to the on-board cluster manager and then have the cluster manager either parse the command string and forward the command(s) to the appropriate satellite(s) or to make some intelligent decision as to the most appropriate action. Two types of ground commanding and controlling are possible. The first type involves sending up a sequence of commands which are intended for specific satellites. With this scenario, one possible implementation method is to use a delimiter to separate commands with each command possessing fields containing the satellite number, command name, value, duration, and time. This description is simplified to illustrate the point and neglects fields such as a CRC field for error checking. This command string would be sent to the on-board cluster manager where the cluster manager would then parse the command string and send commands to the appropriate satellites.

A second type of commanding involves commands which are sent to the cluster without specific indication as to which satellites in the cluster will ultimately be effected. A hypothetical example might be to issue a command to "observe region x at time y". The cluster manager, based on the status of the satellites at time y, will then determine the appropriate course of action to be taken. This type of commanding requires more on-board intelligence than in the first scenario and has a higher level of risk. Because of the higher level of risk, safeguards need to be put in place to ensure no adverse conditions arise.

To command the cluster of satellites a common frequency will be used with a spacecraft ID used to denote what commands are destined for which satellites. The satellites are flying in close enough formation so that they are all within the same beamwidth. All satellites receive the command but not all will process that command. The operation is somewhat analogous to a TCP/IP broadcast system.

Telemetry decommutation on the ground requires being able to parse telemetry from multiple satellites. For a TDM-based telemetry system this scales nicely from how traditional Time Division Multiplexing (TDM)-based systems operate. Telemetry from different satellites simply gets associated with specified frames and frame locations. For CCSDS-based systems packets can contain a field which identifies where they originated.

Developing methods to monitor and visualize cluster state-of-health can be very difficult. Visualizing individual satellite mnemonics can be difficult for moderately complex satellites and this problem gets magnified for a cluster of satellites. One solution is to develop a hierarchical telemetry display system. A top level system would contain the overall status of individual satellites. Choosing an individual satellite and drilling down to the second level would contain a display showing all subsystems for that satellite. Additional levels would partition a subsystem even further. Anomalous conditions would be highlighted at any level by *bubbling* up problems through the hierarchy.

As the core of our ground system we are baselining the *Spacecraft Command Language* (SCL) from Interface and Control Systems. SCL is a Commercial-off-the-Shelf (COTS) software package which contains an expert system and a command scripting language. It was designed to operate both on-board a satellite and on the ground. This makes it an ideal environment for developing a prototype which contains the cluster commanding and monitoring capability described earlier. Expert system rules can be developed and migrated from ground to space as appropriate. Using the rule-based expert system a fault tree for known anomalous conditions can be developed. An

initial ground prototype developed is shown in figure 2. Although this is for a simplified example it provides cluster level state-of-health monitoring and control. A prototype being developed is adding the hierarchical monitoring capability.

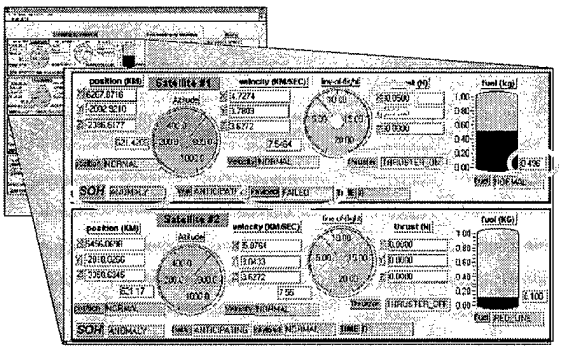


Figure 2: Ground station display

SCL is also being used to handle all commanding and telemetry between the ground and the cluster and to simulate the on-board Command and Data Handling (C&DH) system. This will be described in section 4.

2.2 FLIGHT SYSTEM

This section discusses the flight system portion of the testbed. First an overview of the ObjectAgent and TeamAgent systems is given. ObjectAgent and TeamAgent are two of the key components of the flight system as it pertains to distributed satellite command and control. This is followed by descriptions of the flight system architecture and the cluster manager.

OBJECTAGENT AND TEAMAGENT

ObjectAgent and TeamAgent are being developed by Princeton Satellite Systems with phase II funding from the Air Force Research Laboratory (AFRL) under the Small Business Innovative Research (SBIR) Program. ObjectAgent is an agent based, message passing architecture for use with distributed systems. TeamAgent applies the ObjectAgent architecture to constellations of multiple cooperating satellites. Thus, both systems are well suited for use in the command and control of satellite clusters.

Agents in ObjectAgent can be used at all levels of software functionality because there is no set level of complexity for an agent. For instance, a designer could choose to implement the entire flight software as a single agent, he could use one agent for the software related to each subsystem, or he could use multiple agents for each subsystem etc. ObjectAgent agents are composed of skills. The skills that an agent possesses determine its complexity and functionality. However, all agents have some basics skills to ensure that they can communicate. In addition, agents have self-knowledge and they can explain their functioning and purpose to other agents and users. Agent communication takes place solely through messages, there is no shared memory between agents. This ensures that agents can work together even when they are not located on the same processor.

A TeamAgent demonstration using the ObjectAgent architecture prototyped in Matlab was conducted in January 2000. The demonstration showed the ability to control a cluster of satellites using agents. A cluster of four satellites was simulated. The satellites were placed in an elliptical trajectory relative to a reference orbit. The payload of one of the satellites was then failed. The failed satellite was removed from the cluster, and the remaining three satellites were repositioned along the elliptical trajectory to compensate for the removed satellite. The cluster was controlled in a leader/follower fashion with one of the satellites in charge of determining a need to reconfigure as well as of calculating reconfiguration trajectories. The same satellite then issued thrust commands to the remaining satellites. A more detailed account of this demonstration can be found in [6][7][8].

The initial work for ObjectAgent™ and TeamAgent™ was done in Matlab. Full details of this work can be found in [4][9]. At present, the ObjectAgent architecture is being ported to C++ for implementation on the OSE™ real-time operating system (RTOS). Future TeamAgent demonstrations will use the C++/OSE version of ObjectAgent. More details regarding the status of the C++ version of ObjectAgent are given in [9].

FLIGHT SYSTEM ARCHITECTURE

The flight system section of the testbed consists of eight Force PowerCore 6750 boards. The boards have a single PowerPC 750 processor and are housed in a VME chassis. They are connected using 100 Mbps ethernet. In addition each board has two RS-232 interfaces. Each board is running Enea's OSE RTOS. OSE is a message passing operating system well suited for distributed applications.

As can be seen in Figure 1, the flight system interfaces with a simulation environment and with the ground segment of the testbed. The simulation includes spacecraft dynamics, environmental factors, and actuator and sensor models. It provides inputs to the software on the flight boards and receives software outputs to the spacecraft actuators. At present, the simulation is connected to each board via one of its serial interfaces. In the future, the boards will interface with the simulation environment through ethernet. Communication between the ground and flight systems is accomplished through the ethernet and is handled by SCL, which is present at both ends of the interface. This interface

will be discussed further in the section describing prototype development.

The testbed can be used to simulate a cluster of up to eight satellites with the flight software for a single satellite running on each of the processing boards. The prototype system being tested at the present time, however, consists of a three satellite cluster organized in a leader/follower fashion. The leader satellite is known as the cluster manager, it carries software to allow it to make cluster level decisions and to issue commands to the follower satellites. Because the two follower satellites in this system are designated as primary and secondary back-ups to the cluster manager, they carry the same software on-board; however, software pertaining to cluster manager functionality is turned off until the satellite needs to assume the function of cluster manager.

THE CLUSTER MANAGER

As mentioned above, the cluster manager is responsible for making cluster level decisions. It is this piece of software which allows the satellite cluster to function as a "virtual" satellite. The cluster manager functionality is broken down into the following four major areas:

- Command and control
- Cluster data management
- Formation flying
- Fault management

The command and control portion of the cluster manager is implemented using a combination of SCL, Casper and ObjectAgent. Intersatellite communication and communication between the cluster manager and the ground is implemented using SCL. Casper is used to help break down high level commands into lower level commands and to help plan implementation of complex tasks. Both SCL and ObjectAgent are used to generate commands for other spacecraft, depending on the type of algorithm generating the command. The capabilities provided by the cluster manager command and control allow the cluster to be treated as a "virtual" satellite.

Cluster data management is needed because the cluster must be able to provide state of health information for all the satellites in the cluster. In addition, it must keep track of data, such as relative position and velocity, needed to control the cluster. Potentially, the cluster must be able to provide any telemetry data requested by the ground for any of the satellites in the cluster. The SCL database is used to keep track of all necessary data. Some data is provided by the satellites to the cluster manager on a periodic basis, while other data is only provided upon request. The information that the cluster manager keeps in its database is still to be determined, but at a minimum it includes the following (for each satellite in the cluster):

- Relative position
- Relative velocity
- Absolute position
- Absolute velocity
- Attitude quaternion
- System time
- Spacecraft mode
- Fuel level
- Reference trajectory
- Sensor states

The formation flying part of the cluster manager is responsible for maintaining the cluster formation and for reconfiguring the cluster whenever necessary. The algorithms necessary for formation flying are implemented as ObjectAgent agents. The inputs needed by the agents are sometimes provided by other agents and sometimes obtained from the SCL database. The outputs from the formation flying segment of the cluster manager are commands for the members of the cluster. Thus, the formation flying portion of the cluster manager is implemented using ObjectAgent but it has strong interfaces to both the cluster data management and the command and control portions.

The cluster manager will be responsible for identifying and handling cluster level faults. Cluster level faults are those faults which require action from the cluster in order to be managed. An example of a cluster level fault is a failure of the intersatellite link in one of the satellites. In this case, though the spacecraft in question may still be able to function as an individual satellite, it is no longer able to participate as a member of the cluster and the cluster manager must compensate for this fact. Cluster level fault management will be implemented using a combination of ObjectAgent and SCL. At present more details cannot be provided because the fault management portion of the cluster manager has not been fully defined.

3. PROTOTYPE DEMONSTRATION

Several prototypes have been or are currently under development which show initial operation of our cluster management system.

The initial prototype, which was Matlab based, included four satellites flying in formation which communicated to an SCL based ground station via sockets. Within the prototype the system had the ability to: command the cluster, receive telemetry from the cluster, display cluster status, inject faults such as a GPS failure, perform autonomous reconfiguration, and fly in formation using Hill's equations. Upon GPS failure the on-board cluster manager would autonomously remove the failed satellite from the cluster. Orbital data was sent down to the ground station and forwarded to a graphical

display system where operation could be visualized in real-time.

A second series of prototypes being developed extends the above system and implements the on-board cluster management system on the PowerPC architecture described earlier. As mentioned earlier, SCL is used as the on-board Command and Data Handling (C&DH) system and will handle all communication between the on-board cluster manager and the ground system. The communication protocol was CCSDS. The initial prototype contained approximately 15-20 telemetry points and allowed for ground commanding. The initial system showed a prototypical on-board C&DH system but was not integrated with TeamAgent. An environmental simulator, which interfaced with the PowerPC's, was used to generate orbital data. A second prototype under development will show integration of the TeamAgent system with the C&DH subsystem. The ground system developed in the initial prototype contained several features which will be enhanced in future prototypes. This included: telemetry decommutation, a graphical fault tree, telemetry display, commanding ability, web-based fault diagnosis. In addition the ground system had an interface with Satellite Toolkit for displaying satellite orbits.

4. Conclusion / Future Research

At the time of this writing a cluster management demonstration is being developed which essentially migrates the initial cluster manager described earlier and implemented in Matlab to the PowerPC architecture. This demonstration will show autonomous reconfiguration, a limited ACS implementation, as well as collision avoidance.

Subsequent prototypes will be developed which will enhance cluster management functionality and eventually lead to implementation of an actual cluster management system for TechSat-21 which will contain both flight and ground components.

Flying a cluster of satellites in tight formation requires an on-board cluster management system in order to reduce response time, provide fault tolerance, and enhance mission performance. In addition a ground system capable of interacting with the flight system is also required in order to offer failsafe operation and to reduce ground manpower requirements and costs. The cluster management system being development in the AFRL testbed is well on the way towards achieving our objectives.

5. References

[1] "TechSat 21: Advanced Research and Technology Enabling Distributed Satellite Systems", *Overview Briefing of TechSat 21*, http://www.vs.afrl.af.mil/vsd/techsat21.

[2] Wyatt J., Sherwood R., Sue M., Szijjarto J., "Flight Validation of On-Demand Operations: The Deep Space One Beacon Monitor Operations Experiment", *Proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, ESTEC, The Netherlands, June 1999.

[3] Zetocha P., Self L., "An Overview of Agent Technology for Satellite Autonomy", *1999 FLAIRS Conference Proceedings*, Orlando FL, May 1999.

[4] Paluszek M., Thomas S., Sullivan W., Surka D., "ObjectAgent System Architecture for Autonomous Spacecraft", *Phase I SBIR Final Report*, May 1999.

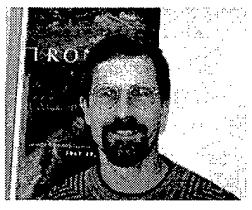
[5] Zetocha, P., "Satellite Cluster Command and Control", *Proceedings of the IEEE Aerospace 2000 Conference,* Big Sky MT, Mar 2000.

[6] Schetter, T. P., M. E. Campbell, and D. M. Surka, "Comparison of Multiple Agent-based Organizations for Satellite Constellations," *2000 FLAIRS AI Conference*, Orlando, Florida, May 2000.

[7] Schetter, T.P., M. E. Campbell, and D. M. Surka, "Multiple Agent-Based Autonomy for Satellite Constellations," *Second International Symposium* on Agent Systems and Applications, Zurich, Switzerland, September 2000.

[8] Surka, D. M., M. C. Brito, and C. G. Harvey, "Development of the Real-Time ObjectAgent Flight Software Architecture for Distributed Satellite Systems," To be Presented at *IEEE Aerospace Conference*, Big Sky, Montana, March 20001.

[9] Surka D, Brito M, Paluszek M., Schetter Thomas, Campbell M., "The TeamAgent System for Multiple Satellite Constellations", *Phase I SBIR Final Report*, April 2000..



Paul Zetocha is a Computer Engineer and is currently the lead for the Intelligent Satellite Systems Group of the Air Force Research Laboratory's Space Vehicles Directorate. For the past eight years he has been involved, both as Program Manager and through in-house development, with over a dozen programs related to spacecraft autonomy. He is also the Chairman of the AIAA Intelligent Systems Technical Committee. Mr. Zetocha has received M.S. degrees in both Electrical Engineering and Computer Science from the University of New Mexico with an emphasis in the areas of signal processing and artificial intelligence respectively.

Margarita Brito is an Aerospace Engineer with Princeton Satellite Systems. Ms. Brito joined the company in September 1999 and has been in New Mexico since April 2000. She is working with others to develop ObjectAgent software to run on the OSE Real Time Operating System. In addition, she is responsible for the integration of ObjectAgent software into the AFRL TechSat 21 Testbed. Ms. Brito is also responsible for the integration of an attitude propagator with the AFRL Testbed.